ITKA203 Käyttöjärjestelmät -- tenttitärpit -- kevät 2014

Yleistä tentistä ja tärpeistä

Jokaisessa kysymyksessä on pistemäärä, jonka arvoiseksi se laajuudeltaan katsotaan. Jokaiseen tenttikertaan tullaan valitsemaan tässä lueteltujen tehtävien joukosta erilaisia siten että pistesumma on 24. Tarkastus jaetaan luennoitsijan ja tuntiopettajien kesken tehtävittäin, jotta kunkin tehtävän sisäinen arviointi on yhtenäinen opiskelijoiden kesken. Tästä syystä **vastaa tehtäviin/tehtäväryhmiin eri vastauspapereille kysymyspaperissa olevien ohjeiden mukaisesti**, jotta tarkastajien ei tarvitse vaihdella papereita keskenään. **Muista myös laittaa oma nimi ja syntymäaika jokaiseen vastauspaperiin ja merkitä tehtävänumerot selkeästi**.

Aihepiirit jakautunevat tentissä suurin piirtein seuraavasti: 4-6 pistettä laitteiston toimintaa ja konekieliohjelmointia, 6-8 pistettä shellin, unix-komentojen ja skriptien käyttöä ja loput 12 pistettä käyttöjärjestelmän ytimen eri osa-alueisiin, yleisiin periaatteisiin ja historiaan liittyviä kysymyksiä.

Arvosana läpäisyrajalla eli 12 pisteellä on 1 ja täysillä pisteillä 5. Pistealue 12-24 jaetaan tasaväleihin neljällä väliarvolla, jotka pyöristetään lähimpään 0.5 pisteen granulariteetilla ilmaistavaan pistemäärään:

Pisteväli	**Välin pituus**	**Arvosana**
[0.0, 12.0)	12p	hylätty
[12.0, 14.5)	2.5p	1
[14.5, 16.5)	2.0p	2
[16.5, 19.0)	2.5p	3
[19.0, 21.5)	2.5p	4
[21.5, 24.0]	2.5p	5

Jos pistemäärä osuu täsmälleen rajalle, luetaan arvosana rajan paremmalta puolelta, esim. 19.0 pistettä riittää arvosanaan 4.

Vapaaehtoisista demoista saadut bonuspisteet huomioidaan **vain siinä tapauksessa, että tenttivastauksista kertyy vähintään tuo 12 pistettä**. Bonuspisteet ovat voimassa kesän 2014 uusinnoissa. Kurssimerkinnän saaminen edellyttää, että pakolliset demot on tehty ja palautettu. **Osasuoritusten hyväksiluvusta myöhemmillä kurssikerroilla ei ole eikä tule mitään lupauksia**. Jotta ensimmäiseen tenttipäivään ja uusintapäiviin osallistuvat opiskelijat olisivat arvostelun osalta keskenään samanarvoisessa asemassa, **bonuspisteitä saa vain takarajaan 31.5.2014 mennessä palautetuista vapaaehtoisista demoista**. Arvosanaa ei siis voi hinkuttaa ylöspäin jälkikäteen. "Matti- ja Maijamyöhäset" voivat kuitenkin perustellusta syystä palauttaa *pakollisia* demoja eli 1-4 aina kesän viimeiseen

uusintaan asti; tämä ei käsittääkseni vaikuta arvosteluun mitenkään eriarvoistavasti. Kesäkuun alusta alkaen palautuksista tulee sopia etukäteen meilillä tai puhelimitse. Kevätkurssin Optima-ympäristön jätämme ilolla taaksemme 1.6.2014.

Jotkut tentin tehtävät ovat luonteeltaan muuttuvaisia: Koodit (C- ja bash-ohjelmat sekä pseudokoodit), joista kysymykset tehdään, voivat olla erilaiset mutta vaativuudeltaan vastaavat kuin tässä on esimerkinomaisesti annettu. Niihin pystyy vastaamaan, jos on tutustunut kirjalliseen materiaaliin (mukaanlukien pakolliset demot) sekä esiteltyihin esimerkkiohjelmiin. Soveltamiskykyä (toivottavasti) jonkin verran edellytetään.

Jotkut tehtävät ovat yleisessä "esseevastausta odottavassa" muodossa. Toisaalta tekisi mieli kysyä vain "helposti tarkastettavia" lyhyen ja täsmällisen vastauksen kysymyksiä, mutta toisaalta niihin ei mahtuisi yhtään pohdintaa, joka sinänsä on tärkeää. Mahdollisessa esseessä pyri vastaamaan ytimekkäästi seuraaviin asioihin: (1) Miksi kysyttävä asia on tärkeä, eli mitä sillä saavutetaan. (2) Mitä asiaan liittyy käytännön tasolla, eli esim. jokin sovellusesimerkki on hyvä tapa osoittaa ymmärrys tästä. (3) Miten asian voi käyttöjärjestelmässä pääpiirteittäin toteuttaa, eli esittele tietorakenteet ja riittävän tarkat pseudokoodit.

Kysymysten jaottelu mukailee tässä kevään 2014 kirjallisen materiaalin otsikointia.

Sisältö

Yleisiä kysymyksiä koko alueelta

Kysymys:

Selitä, mitä seuraavat termit tarkoittavat tietokonelaitteiston ja käyttöjärjestelmän yhteydessä:

a) X1
b) X2
...
f) X6

(max. 6p; 1p/termi)

Tässä X1, ..., X6 ovat luentomonisteessa lihavoidulla kirjasimella merkattuja avainsanoja. "Pieni greppaus" antoi vuonna 2014 seuraavanlaisen listan, josta konekieleen ja ohjelmointikieliin liittyvät yksityiskohdat on suodatettu käsipelillä pois:

rajapinta tietokone bitti logiikkaportti keskusyksikkö prosessori I/O-laite väylä osoiteavaruus tavu sananpituus sana ROM-muisti RAM-muisti kontrolliyksikkö "aritmeettislooginen yksikkö" rekisteri "nouto-suoritus -sykli" "(käskyn) nouto" "(käskyn) suoritus" "(konekieli-)käsky" konekieli käskyosoitin ohjelmalaskuri operandi suoritus lippurekisteri järjestelmärekisteri "käyttäjälle näkyvä rekisteri" käyttöjärjestelmätila kuori ikkunointijärjestelmä työpöytä käyttäjätila prosessoriarkkitehtuuri operaatiokoodi käskykanta käskykanta-arkkitehtuuri välimuisti lokaalisuusperiaate massamuisti muistihierarkia "symbolinen konekieli" assembler käskysymboli disassembly suorituspino aliohjelma "kontrollin siirtyminen" koodi(-alue) data(-alue) "paikallinen muuttuja" pino(-alue) kekomuisti viitemuuttuja virtuaaliosoite "segmentoitu muisti" segmenttirekisteri vuoronnus aikataulutus työ eräajo käyttäjätila valvontatila keskeytys moniajo moniohjelmointi aikajakojärjestelmä prosessi keskeytyspulssi keskeytyskäsittely FLIH kernel-pino keskeytysvektori "ohjelmallinen keskeytyspyyntö" "(käyttöjärjestelmän) kutsurajapinta" prosessi konteksti "kontekstin vaihto" prosessitaulu prosessielementti vuorontaja kiertojono säie "kevyt prosessi" signaali viesti "jaettu muistialue" putki postilaatikko portti etäaliohjelmakutsu semafori "kriittinen alue" "atominen operaatio" deadlock nälkiintyminen lukkiutuminen "(virtuaalimuistin) sivu" "(sivu-)kehys" "työjoukko" sivutaulu kehystaulu sivunvaihtokeskeytys LRU laiteohjain sovitin kanava kontrollilogiikka "(kovalevyn) ura" "(kovalevyn) sektori" "(kovalevyn) sylinteri" hakuaika kovalevyn vuoronnus reaaliaikajärjestelmä pre-emptiivisyys

Lisäksi seuraavat pakollisissa demoissa ja Petterin vierailuluennolla käydyt on osattava:

shell "interaktiivinen shell" (shell-)skripti virtuaalikone virtuaalipalvelin RAID

Vastauksena ei odoteta romaania, vaan lyhyt ja ytimekäs, yhden pisteen arvoinen selvitys jokaista selitettävää avainsanaa kohden. Tähän "kaatopaikkatehtävään" päätynee pääasiassa sellaisia avainsanoja, joita muissa (kenties syvällisemmissä) tehtävissä ei käsitellä.

Kysymys:

(Petterin vierailuluennolta):

a) Mitä tarkoittaa palvelininfrastruktuurin virtualisointi?

b) Mitä hyötyä virtualisoinnista on verrattuna erillisiin palvelinkoneisiin?

c) Mitä tarkoittaa NAS (network attached storage) ja mitä sillä saavutetaan?

(3p)

Tietokonelaitteisto

Laitteistoarkkitehtuuri on "esitieto" eikä sinänsä Käyttöjärjestelmät-kurssin asia, eli laitteistosta sinänsä ei tulla kysymään paljonkaan. Kuitenkin ymmärrät laitteiston rakenteen ja käsitteet (materiaalissa boldatut sanat), joita käytät monissa vastauksissasi, jos ne ovat niin sanotusti hyviä vastauksia. Seuraavat kysymykset ovat kuitenkin sisällön kannalta olennaisia:

Kysymys:

a) Mitä rekisterejä prosessorissa täytyy vähintään olla?

b) Mihin kutakin näistä välttämättömistä rekistereistä käytetään?

c) Mitä muita muistikomponentteja tietokoneessa on, ja mitkä näiden merkittävät erot ovat (ts. selitä "muistihierarkia")?

(1+1+2 = 4p)

Kysymys:

(Erilainen tapa kysyä; tuunaa tenttisi -demon vastauksesta):

a) Listaa muistikomponentit nopeasta hitaaseen.

b) Miksi tarvitaan näin monia erilaisia muistikomponentteja eikä riitä vain yhdenlaiset?

(2p)

Kysymys:

a) Missä toimintatiloissa prosessorin täytyy vähintään pystyä olemaan?

b) Miksi?

c) Miten tämä vaikuttaa prosessorin käskykannan ja rekisterien käyttöön?

(1+1+1 = 3p)

Konekielisen ohjelman suoritus

Kysymys:

a) Esitä prosessorin nouto-suoritussykli (mukaanlukien keskeytyskäsittely) kaaviona
b) selitä lyhyesti, mitä syklin missäkin vaiheessa tapahtuu.

(2p)

Kysymys:

a) Mihin eri alueisiin (käyttötarkoituksen mielessä) ohjelman tyypillisesti käyttämä muisti jakaantuu?
b) Selitä, mikä on virtuaalimuistiavaruus ja miten se eroaa fyysisestä muistiavaruudesta.
c) Selitä, mikä on "osoitin" ja mitä se yksinkertaisimmillaan tarkoittaa konekielessä.

(3p)

Käyttöjärjestelmän yleiskuva

Kysymys:

Kuvaile niitä tietokonelaitteiston käytön ongelmia, joiden ratkaisemiseksi nykyaikaisen käyttöjärjestelmän ominaisuudet ovat historian saatossa syntyneet? Minkä alijärjestelmän/moduulin päätehtävä kunkin ongelman ratkaiseminen nykyaikaisessa käyttöjärjestelmässä on?

(3p) [Riittävässä 3p vastauksessa on kuvailtava mm. syyt moniajolle/vuorontamiselle, virtuaalimuistille, useampikerroksiselle I/O -ohjelmistolle ja monen käyttäjän oikeuksien hallinnalle!]

Kysymys:

Mihin tarvitaan keskeytyksiä?

(1p)

Kysymys:

Selvitä keskeytyskäsittelyn perusmekanismeja:

a) Miksi ylipäätään keskeytyksiä tarvitaan?
b) Kerro kolme esimerkkiä keskeytyksestä: mistä keskeytys tuli, miksi, ja millä tavoin?

c) Olipa syy keskeytykseen mikä tahansa, niin mitä prosessori tekee keskeytyksen tultua (kuvaile siis FLIHin tärkeimmät osuudet)?
d) Selitä jokin käyttöjärjestelmän keskeytyskäsittelytoimenpide (sopivalla tarkkuudella, joko sanallisesti tai pseudokoodina).

(4p)

Kysymys:

Luettele (nimeltä) kahdeksan käyttöjärjestelmän palvelua, joita kutakin voisi vastata yksi käyttöjärjestelmäkutsu.

(2p)

Kysymys:

Miten käyttöjärjestelmän palveluiden käyttö tapahtuu? Eli miten palvelun käyttö näyttäytyy:

a) sovellusohjelmoijan lähdekoodissa
b) prosessorilaitteiston toimenpiteenä
c) käyttöjärjestelmän toteutuksessa.

(2p)

Kysymys:

Anna esimerkki käyttöjärjestelmän toteutuksessa tai konfiguroinnissa välttämättömästä kompromissista, eli kahdesta ristiriitaisesta toiminnallisesta tai laadullisesta tavoitteesta, joita ei voida saada yhtä aikaa optimaalisiksi.

(2p)

Kysymys:

Mitä ovat käyttäjätila ja käyttöjärjestelmätila, ja miksi ne tarvitaan?

(2p)

Prosessi ja prosessien hallinta

Kysymys:

Mitä käyttöjärjestelmän yhteydessä tarkoitetaan sanalla "prosessi"?

(1p)

Kysymys:

Mikä on käyttöjärjestelmän prosessitaulu ja mitä se sisältää?

(2p)

Kysymys:

a) Mikä on prosessielementti (PCB)?
b) Kuka tai mikä käyttää prosessielementtiä ja mihin?
c) Mitä prosessielementti sisältää?

(2p)

Kysymys:

Voit käsitellä luennoilla tai materiaalissa esitetyssä laajudessa, oppikirjan esitteleminä konkreettisina esimerkkeinä, tai siinä laajuudessa kuin käsittääksesi on välttämätöntä järkevän moniajon toteutumiseksi:

a) Missä eri tiloissa prosessi voi olla?
b) Milloin tapahtuu mikäkin siirtymä tilojen välillä?

(1+2 = 3p)

Kysymys:

Täydennä seuraavan "pseudo-C-kielisen" ohjelman kommentit. Aliohjelma `luekomento()` lukee arvot parametreilleen, `pid` on kokonaisluku:

```
/* Tämä on minimalistinen esimerkkitoteutus ohjelmasta, jollaista
 * tyypillisesti nimitetään ... [VASTAUKSESI a-kohta].
 *
 * Ohjelman avulla sen käyttäjä voi ... [VASTAUKSESI b-kohta]
 */

while(true){
  luekomento(komento, parametrit);
  pid = fork();    /* pid:hen sijoitetaan fork()-funktion
                    * paluuarvo. Tuo kyseinen fork() vastaa
                    * UNIX-järjestelmässä käyttöjärjestelmäkutsua,
                    * jonka tehtävä on ... [VASTAUKSESI c-kohta].
                    */
  if (pid > 0) {
```

```
    /* Suorituksen saapuessa tähän lohkoon, tiedetään
     * käyttöjärjestelmän tilasta, että ... [VASTAUKSESI d-kohta]
     * ja tästä prosessista, että ... [VASTAUKSESI e-kohta]
     */
    status = wait(); /* Tämän rivin käyttöjärjestelmäkutsun wait()
                      * tehtävä on ... [VASTAUKSESI f-kohta]
                      */
  } else if (pid == -1) {
    /* Suorituksen saapuessa tähän lohkoon, tiedetään
     * käyttöjärjestelmän tilasta, että ... [VASTAUKSESI g-kohta]
     * ja tästä prosessista että ... [VASTAUKSESI h-kohta]
     */
    exit(1)         /* Tämän rivin käyttöjärjestelmäkutsun exit()
                     * tehtävä on ... [VASTAUKSESI i-kohta]
                     */
  } else {
    /* Suorituksen saapuessa tähän lohkoon, tiedetään
     * käyttöjärjestelmän tilasta, että ... [VASTAUKSESI j-kohta]
     * ja tästä prosessista että ... [VASTAUKSESI k-kohta]
     */
    exec(komento, parametrit);
    /* Edellisen rivin käyttöjärjestelmäkutsun exec() tehtävä on
     * ... [VASTAUKSESI l-kohta]
     */
    /* Niinpä ollen seuraavan rivin suoritus
     * ... [VASTAUKSESI m-kohta]
     */
    printf("%d",pid);
  }
}
```

(4p)

Kysymys:

a) Mikä on säikeen ja prosessin ero?

b) Missä tilanteessa käyttäisit mieluummin säiettä kuin prosessia?

(2p)

Kysymys:

Tässä on komennon `ps` antama tuloste:

```
UID        PID  PPID  C STIME TTY          TIME CMD
nieminen  1053     1  0 Apr15 ?        00:00:22 SCREEN
nieminen  1054  1053  0 Apr15 pts/47   00:00:00 /bin/bash
nieminen  4194  1053  0 Apr24 pts/59   00:00:00 /bin/bash
```

```
nieminen 12561 26431  0 Apr22 pts/139  00:00:11 emacs kj14/luennot.html
nieminen 13461 16562  0 11:06 pts/22   00:00:00 screen -d -r
nieminen 13466  1054  0 11:06 pts/47   00:00:00 pine
nieminen 16561 16552  0 May07 ?        00:00:07 sshd: nieminen@pts/22
nieminen 16562 16561  0 May07 pts/22   00:00:00 -bash
nieminen 22315 22303  0 15:07 ?        00:00:00 sshd: nieminen@pts/221
nieminen 22316 22315  0 15:07 pts/221  00:00:00 -bash
nieminen 22396 22316 19 15:08 pts/221  00:00:00 ps -f -u nieminen
nieminen 26431  1053  0 Apr22 pts/139  00:00:00 /bin/bash
```

a) Selitä sarakkeiden `UID`, `PID`, `PPID` ja `CMD` merkitys suomen kielellä.

b) Mistä prosessista käynnissä oleva tekstinkäsittelyohjelma todennäköisesti on haaroitettu eli forkattu?

(2 + 1 = 3p)

Yhdenaikaisuus, prosessien kommunikointi ja synkronointi

Kysymys:

Luettele (nimeltä) neljä erilaista prosessienvälisen kommunikoinnin (IPC) menetelmää.

(1p)

Kysymys:

Mitä ovat signaalit, ja miten niitä käytetään?

(2p)

Kysymys:

Miten tapahtuu viestinvälitys send() ja receive() -käyttöjärjestelmäkutsuilla?

(2p)

Kysymys:

Seuraavaa aliohjelmaa voidaan mahdollisesti suorittaa useammassa kuin yhdessä säikeessä, ja viestikanava `statuskan` on säikeille yhteinen. Toistaiseksi kukaan ei ole miettinyt synkronointikysymyksiä ohjelmistossa, johon aliohjelma liittyy.

a) Millainen ongelma tässä voi tulla, ja missä tilanteessa?

b) Kerro, miten käyttäisit semaforia ongelman ratkaisemiseen (koodia ei ole pakko kirjoittaa, kunhan selität ratkaisuidean).

Koodi:

```
/* Tulostaa toimintatilan statuskanavaan tekstinä
 * Käytämme tasan 10 merkin mittaisia statuskoodeja,
 * joiden perusteella ydinvoimalamme koneisto säätyy
 * uuteen tilanteeseen.
 */
tulosta_status(char teksti[]){
  int i;
  for (i=0;i<10;i++){
    fputc(teksti[i], statuskan); /* merkin tulostus kanavaan */
  }
}
```

(1+2 = 3p)

Kysymys:

a) Millainen tietorakenne on semafori (siis mitä tietoja se sisältää)?

b) Nimeä (tai kuvaile, jos et muista nimiä) kaksi erilaista tarkoitusta, joihin semaforia voi käyttää.

c) Miten on toteutettu kaksi tärkeintä semaforiin liittyvää palvelua eli `wait()` ja `signal()`? Mieluiten kirjoita pseudokoodi; vähintäänkin kuvaile toimenpiteet tarkoin.

(2p)

Kysymys:

a) Selitä käsite "kriittinen alue".

b) Selitä käsite "atominen toimenpide".

(2p)

Kysymys:

a) Selitä poissulkuongelma: millaisissa tilanteissa se voi tulla eteen, ja mitä se esimerkiksi voi käsittelemättömänä aiheuttaa.

b) Esitä C-ohjelma tai pseudokoodi, jossa poissulku tehdään käyttäen käyttöjärjestelmän semaforipalveluita.

(2p)

Kysymys:

a) Kuvaile tuottaja-kuluttaja -ongelma.

b) Esitä C-ohjelma tai pseudokoodi, jossa tuottaja-kuluttaja -ongelma hoidetaan käyttäen käyttöjärjestelmän semaforipalveluita.

(3p)

Kysymys:

a) Anna esimerkki tilanteesta, jossa voi tapahtua lukkiutuminen eli deadlock -tilanne.

b) Täsmennä, missä olosuhteissa esimerkissäsi tapahtuu lukkiutuminen.

(2p)

Kysymys:

a) Selitä lukkiutumisen ja nälkiintymisen ero.

(2p)

Muistinhallinta

Kysymys:

Tässä riittää käsittely kurssimateriaalin kuvailemassa laajuudessa (jos tiedät enemmän, se on hyvä, mutta älä yritä kirjoittaa sitä novellia tähän).

a) Mihin nykyaikaisen käyttöjärjestelmän ja laitteiston muodostamaan kokonaisuuteen liittyvät käyttöjärjestelmän tietorakenteet nimeltä "sivutaulu" ja "kehystaulu"?

b) Mikä tämän kokonaisuuden tavoite on? Siis miksi se on kehitetty?

c) Mikä on sivutaulu, ja mitä tietoja se sisältää?

d) Montako sivutaulua täytyy vähintään olla olemassa kullakin hetkellä?

e) Mikä on kehystaulu, ja mitä tietoja se sisältää?

f) Montako kehystaulua täytyy vähintään olla olemassa kullakin hetkellä?

g) Kuvaile, missä tilanteissa ja mihin tarkoituksiin näiden eri taulujen eri tietoja tarvitaan?

(.5 + .5 + 1 + .5 + 1 + .5 + 1 = 5p)

Kysymys:

Selitä (pseudokoodina tai sanallisesti), miten käyttöjärjestelmä käsittelee sivunvaihtokeskeytyksen (page fault). Oleta että kehyksen valintamenetelmänä on LRU (least-recently-used). Kerro myös, mitä sisäisiä tietorakenteita käyttöjärjestelmä hyödyntää ja miten niitä muutetaan.

(3p)

Oheislaitteiden ohjaus

Kysymys:

Merkin lukeminen päätteeltä on esimerkki I/O -operaatiosta. Kerro, miten luku tapahtuu: mitkä kaikki ohjelmisto- ja laitteisto-osat osallistuvat sen toteuttamiseen missäkin vaiheessa, ja mitä niiden kunkin vastuulla on? Aloita kertomus hetkestä, jolloin käyttäjän ohjelma tekee operaatiopyynnön, ja lopeta siihen, kun operaatio on valmis ja käyttäjän ohjelma jatkuu taas seuraavasta käskystä. (Tyyli vapaa - kaaviot, pseudokoodit, essee käyvät).

(3p)

Kysymys:

Megatavun mittaisen JPG-kuvan lukeminen kovalevyltä muistiin myöhempää dekompressointia ja tulostamista varten on esimerkki I/O -operaatiosta. Kerro, miten luku tapahtuu: mitkä kaikki ohjelmisto- ja laitteisto-osat osallistuvat sen toteuttamiseen missäkin vaiheessa, ja mitä niiden kunkin vastuulla on? Aloita kertomus hetkestä, jolloin käyttäjän ohjelma tekee operaatiopyynnön, ja lopeta siihen, kun operaatio on valmis ja käyttäjän ohjelma jatkuu taas seuraavasta käskystä. (Tyyli vapaa - kaaviot, pseudokoodit, essee käyvät).

(3p)

Kysymys:

Miksi tarvitaan erikseen laiteriippuva ja laiteriippumaton I/O -ohjelmisto?

(1p)

Tiedostojärjestelmä

Kysymys:

Millaisia tiedostokohtaisia tietoja tiedostojärjestelmän tietorakenteissa (esim. unixin i-solmut) on pidettävä yllä?

(2p)

Kysymys:

Mitä tarkoittaa unixin iskulause "kaikki on tiedosto"?

(1p)

Kysymys:

Mikä on RAID? Miksi / mihin tarkoitukseen sitä käytetään? Mihin RAIDin toiminta perustuu?

(2p)

Kysymys:

Laitoin talon grillijuhlien digikuvat nettiin naapureita varten, mutta he sanovat että "kuvat eivät toimi". Hakemistolistaukseni näyttää tältä:

```
-bash-2.05b$ ls -l ~/www/grillikuvat/*.JPG
total 3964
-rw-------    1 nieminen opis      1027107 Jun 9 16:34 IMG_1940.JPG
-rw-------    1 nieminen opis       836088 Jun 9 16:34 IMG_1942.JPG
-rw-------    1 nieminen opis      1101871 Jun 9 16:34 IMG_1943.JPG
-rw-------    1 nieminen opis      1066770 Jun 9 16:34 IMG_1945.JPG
```

Mikä on vialla ja miten voin korjata asian?

(1p)

Käyttöjärjestelmän suunnittelusta

Kysymys:

Luettele (ja selitä lyhyesti) tavoitteita käyttöjärjestelmän suunnittelussa erityisesti prosessorin käyttöön ja vuoronnukseen liittyen.

(2p)

Kysymys:

a) Selitä, mitä tarkoitetaan "reaaliaikajärjestelmällä". Anna esimerkki sellaisesta.

b) Mitä erityisiä tavoitteita reaaliaikajärjestelmälle tyypillisesti asetetaan?

(3p)

Kysymys:

a) Miten microkernel -käyttöjärjestelmä eroaa monoliittisesta?
b) Mitä etuja microkernel -mallissa on?

(2p)

Kysymys:

Mitkä ovat käyttöjärjestelmän päätehtävät?

(1p)

Shellit ja shell-skriptit

TODO: Joitakin "Mitä seuraava komento tekee" -tyyppisiä.

Kysymys:

a) Selitä, mikä on "interaktiivinen shell".
b) Mikä on shell-skripti?

(1p)

Kysymys:

Luettele kolme tarkoitusta, joihin shell-skriptejä voidaan käyttää.

(1p)

Kysymys:

Mitä voi tehdä apuohjelmalla `grep`?

[tai `find`, `whoami`, `ps`, `kill` tai muu monisteessa / pakollisissa demoissa käytelty kaikkein yleisimpiin kuuluva ohjelma]

(1p)

Kysymys:

Olen pääteyhteydessä Linux-palvelimeen, mutta en muista miten unix-komento `whatever` toimii. Miten saan sen selville (tarkistamatta Internetistä)?

(1p)

Kysymys:

Mitä seuraava shell-komentorivi tekee:

```
ls *.jpg > gradu.doc
```

(1p)

Kysymys:

a) Mitä seuraava skripti tekee (hipsukat ovat "backtick"-merkkejä):

```
#!/bin/sh
echo ‘$0‘
```

b) Mikä on lopputulema tämän skriptin ajamisesta?

(1+1=2p) [Vastaus katsotaan kertaus- ja tärppiluennolla (tai jos unohtuu katsoa, niin joutuu jättämään hyvän kysymyksen pois tentistä). Vinkki: ÄLÄ kokeile tätä koneella, jossa on muita käyttäjiä, koska sen jälkeen susta ei tykätä! :-)]

Kysymys:

Mitä seuraava shell-komentorivi tekee (hipsukat ovat "backtick"-merkkejä):

```
grep ‘whoami‘ /etc/passwd > tunnarei.txt
```

Miten sen toiminta eroaa seuraavasta shell-komentorivistä:

```
grep ‘whoami‘ /etc/passwd >> tunnarei.txt
```

(1p)

Kysymys:

Tee bash-skripti, joka testaa onko argumenttina annettu hakemisto olemassa ja oikeasti hakemisto. Esimerkiksi skripti voitaisiin ajaa komennolla:

```
loytyyko hake/alihake
```

Tulostus olisi `kyllä` tai `ei` sen mukaan kumpi on tilanne, ja skriptin lopetuskoodi ilmaisisi "kaikki hyvin" -tilanteen vain silloin kun vastaus on myöntävä. Kieltävän vastauksen yhteydessä lopetuskoodi ilmaisisi "jotakin pielessä" -tilanteen.

(1p)

Kysymys:

Tee bash-skripti, joka lukee päätteeltä yhden rivin tekstiä ja lisää sen tiedoston `~/ajatuksia` loppuun. Skriptin tulee ensin tarkistaa, onko kyseinen tiedosto olemassa ja onko siihen kirjoitusoikeus. Jos ei, niin molemmissa tapauksissa on tulostettava tilannetta kuvaava virheilmoitus ja lopetettava skripti ennen kuin päätteeltä luetaan mitään.

TAI VASTAAVIA SIMPPELEITÄ BOURNE AGAIN SHELL (bash) SKRIPTEJÄ!

(2p)

Kysymys:

Tee pienoinen bash-skripti, jolla saa apuohjelmaa `convert` käyttäen tehtyä kaikista oleskeluhakemistossa olevista .JPG -päätteisistä kuvista 60 pikseliä leveät versiot joiden nimi on mallia `pieni_IMG_1943.JPG` Ei tarvitse tehdä mitään oikeellisuustarkistuksia. Ohje convertin käyttöön:

```
convert -resize 60x LÄHDETIEDOSTO KOHDETIEDOSTO
```

(1p)

Kysymys:

Tee pienoinen bash-skripti, jolla saa muunnettua kaikkien oleskeluhakemistossa olevien `.JPG` -päätteisten tiedostojen nimet päättymään `.jpg` eli pienin kirjaimin. Komento, joka tulostaa argumenttina annetun tiedostonimen alkuosan, toimii seuraavasti:

```
basename .JPG JOTAKIN.JPG
```

Komento, jolla tiedoston nimi muutetaan, on:

```
mv -p ALKUPERAINEN_NIMI  UUSI_NIMI
```

Vinkki: tarkkana hipsukoiden ym. kanssa.

(1p)

Tuunaa tenttisi - poimintoja

Skriptin täydentäminen

Tämä tulee "tuunaa tenttisi" -demon vastauksesta. Kiitos ideasta! Kysymyksessä jokin vastaavantasoinen skripti, joka pitää täydentää.

Esimerkkitehtävänanto suoraan demovastauksesta:

```
Täydennä toimiva scripti a-c kohtien tehtävänannon mukaan

#!/bin/bash -eu
#scripti lukee tekstitiedostosta riveittäin ja
#tulostaa toiseen tekstitiedostoon rivit joiden hakuehto täsmää.
#Mikäli hakuehtoa ei ole annettu kopioidaan suoraan.


a) Toteuta ehtolause, joka tarkastaa onko käyttäjä antanut
   vaaditut parametrit (luettava tied, kirjoitettavan tied. nimi),
   sekä ohjeistaa käyttäjää mikäli kutsussa puutteita. Ohjeistaa
   käyttäjää myös vapaaehtoisten parametrien käyttöön (hakuehto
   (ilman kopio suoraan), --overwrite (antaa luvan kirjoittaa
   olemassa olevaan tiedostoon).

b) varmista, että käyttäjäkomennon suorittaminen on mahdollista
   (tiedostot olemassa, luku, kirjoitusoikeudet jne).

c) toteuta tiedostosta luku, selvitä rivit joihin hakuehto täsmää,
   tulosta täsmäävät rivit tiedostoon

==========================================

Tehtävässä apuna "Yhden sivun sh-luntti".

Oletetaan demojen 1-4 ymmärtäminen, ei muistamista.  Vaaditaan
molempien edellä luettujen soveltamista.  Mittaa mielestäni
pakollisten demojen laajuuden verran scriptin kirjoitustaitoa, ei
kuitenkaan ole aivan demoja vastaava, vaan vaatii yksinkertaista
soveltamista.

- tehtävän voi varmasti suorittaa demojen 1-4 tiedoilla, koska sen
  laatija ei tiedä tippaakaan enempää, mallivastaus myös sen
  näköinen!
```

Arvosteluperusteita (suoraan demovastauksesta; tässä vielä oikeus muutoksiin pidätetään - arvostelu on kuitenkin kaikille sama, oikeudenmukainen ja tarkka):

```
rivin #!/bin/bash -eu voisi mielestäni antaa valmiina, koko muu scripti sitten kirjoitetaankin i
perustelu: minulle ei ainakaan tällä oppimäärällä selvinnyt mitä muuta tuohon voi laittaa
  -> ulkoahan tuon joutuu opettelemaan..

Kuuden pisteen tehtävänä arvosteluesimerkki:

a) max 2p.
1,5p jos:
=======
```

```
if [[ $# -lt 2 ]]
then
JOTAKIN
fi
=======
0,5p: tulostettava neuvo on fiksu ja tulostus menee oikein

b) max 2p.
0,5p: tarkistetaan onko luettava tiedosto olemassa
0,5p: tarkistetaan onko kirjoitettava tiedosto olemassa
0,5p: kahden tarkentimen tapaus -> exit 1
0,5p: neljäs tarkennin != --overwrite -> exit 1
++0,5p jos kolmen tarkentimen tilanne 'luettava kirjoitettava --override' toimii
  (mallivastauksessa c) osiossa)...

osapisteitä jokaisesta toimivasta "osasta", vaikka osaan ei päästäisi muualla olevan virheen vuo

c) max 2p.
1p: rivin luku, hakuehtoon vertaaminen ja tulostus uuteen tiedostoon toimii
    esim. grep $3 < $1 >> $2
1p: huomioitu mahdollisuus ei hakuehtoa -> em. ehtolauseen takana
```

Skriptin korjaaminen

Tehtävän idea: Annetaan skripti, joka selvästi tekee jotakin järkevää, mutta siinä on joitakin virheitä, jotka tulee korjata.

Allaoleva suoraan demovastauksesta; tässä vielä oikeus muutoksiin pidätetään - arvostelu on kuitenkin kaikille sama, oikeudenmukainen ja tarkka):

```
#Kysymys: Seuraavassa on esitetty yksinkertainen skripti, jossa on virheitä.
#         Korjaa skripti kuitenkaan sen rakennetta muuttamatta niin, että se toimii kommenttien

  /bin/sh
pwd > tuloste.txt     # tulostaa nykyisen työskentelyhakemiston tiedoston tuloste.txt loppuun
date > tuloste.txt    # tulostaa päivämäärän tiedoston tuloste.txt loppuun

ls   grep test.sh      # putkita nykyisen kansion tiedostolistaus grep -ohjelmalle ja etsi siitä

if [[ $1 == "" ]]
then
 echo "Ei argumentteja."                   # skripti huomauttaa käyttäjälle, jos sille ei anneta
 echo "Ei argumentteja." > argumentit.txt  # ja luo uuden tiedoston argumentit.txt jonka loppuun
 exit 1                                    # ja lopuksi päättää toimintansa koodilla 1, eli "poi
if

if [[ $1 == "argumentti1" ]]
then
```

```
 echo "Argumentti annettu."                       # jos argumenttina on annettu "argumentti1", sk
 echo "Argumentti $2 annettu." > argumentit.txt  # ja luo uuden tiedoston argumentit.txt, jonka
 exit                                            # ja lopuksi päättää toimintansa koodilla 0, el
fi

# Virheet/lisäykset:
# -> Skriptin riviltä 1 puuttuu "#!"
# -> Skriptin rivillä 3 vain yksi ">", eli tiedoston sisältö korvautuisi vaikka kommentin mukaan
# -> Skriptin rivillä 5 puuttuu merkki "|", joka putkittaa tiedostolistauksen grepille
# -> Skriptin rivillä 10 if rakenne tulee päättyä "fi"
# -> Skriptin rivillä 15 viittaus väärään muuttujaan $2
# -> Skriptin rivillä 16 puuttuu palautettava arvo "0"
```

Liite: Yhden sivun sh-luntti

Alla on tentinkin liitteenä jaettava lunttilappu, jossa esimerkkien kautta muistutetaan joistakin Bourne Shellin piirteistä.

```
Joitakin ohjelmointirakenteita:
    muuttuja=57               # tarkka syntaksista: ei välilyöntejä!
    muuttuja="eki"; muuttuja="$muuttuja$muuttuja"
    read muuttuja             # lukee muuttujaan syöttövirrasta

    if EHTO                   # Myös:  if EHTO; then ...; \
    then                      #        elif EHTO; then ...; fi
      ...                     #
    fi                        #

    for muuttuja in LISTA     # muttujan "esittelyssä"/asetuksessa ei $
    do
      ... jotakin $muuttuja jotakin ... # muuttujan käytössä $muuttuja
    done

    while EHTO; do ... ; ... ; done     # käskyerotin rivinvaihto tai ;

    # Tee jotain syötteen kaikille riveille:
    while read rivi; do echo "luin: $rivi" ; done < rivit.txt

    case $hanen_nimensa in
      kirsimarja)
          echo "Hei Kippe" ;;
      eskomatias)
          echo "Moi E.M." ;;
    esac

    Aliohjelmat mahdollisia&hyödyllisiä, mutta ei käsitellä ITKA203:lla.

Joidenkin ehtojen käyttöä (Välilyönnit merkityksellisiä! "[" on itse
asiassa komento ja loppuosa on sen argumenttilista!):
    [[ -d TIEDOSTONIMI ]]      # tosi, jos on hakemisto
    [[ -f TIEDOSTONIMI ]]      # tosi, jos on tavallinen tiedosto
    [[ -w TIEDOSTONIMI ]]      # tosi, jos on olemassa ja kirjoitettavissa
    [[ ! -f TIEDOSTONIMI ]]    # tosi, jos ei olemassa tai ei tavallinen
    [[ "$muuttuja" -le "57" ]] # tosi, jos muuttuja <= 57 (myös lt,gt,ge)

Argumenttien käyttö:   (yli 9 arg mahd., mutta ei käsitellä ITKA203:lla)
    echo "Tämän skriptin nimi on $0. Eka argumentti $1, neljäs $4"
    echo "Argumenttien määrä on $#"

Joitakin sisäänrakennettuja toimintoja:
    exit VIRHEKOODI              # koodi 0 tarkoittaa onnistumista
    cd                           # vaihtaa skriptin työhakemistoa
    echo                         # kaiuttaa tekstiä
    let muuttuja=$muuttuja+3     # (perusaritmetiikkaa)

Erikoismerkkejä (tuttuja interaktiivisesta käytöstä; toimii skripteissä):
    <   >   >>   |   `KOMENTO`
```